Multilevel Parallelization Architecture of Boundary Element Solver Engines for Cloud Computing

Dipanjan Gope, Don MacMillen,
Devan Williams, Xiren Wang

Outline

- The need for parallelization
- Challenges towards effective parallelization
- A multilevel parallelization framework for BEM: A compute intensive application
- Results and discussion

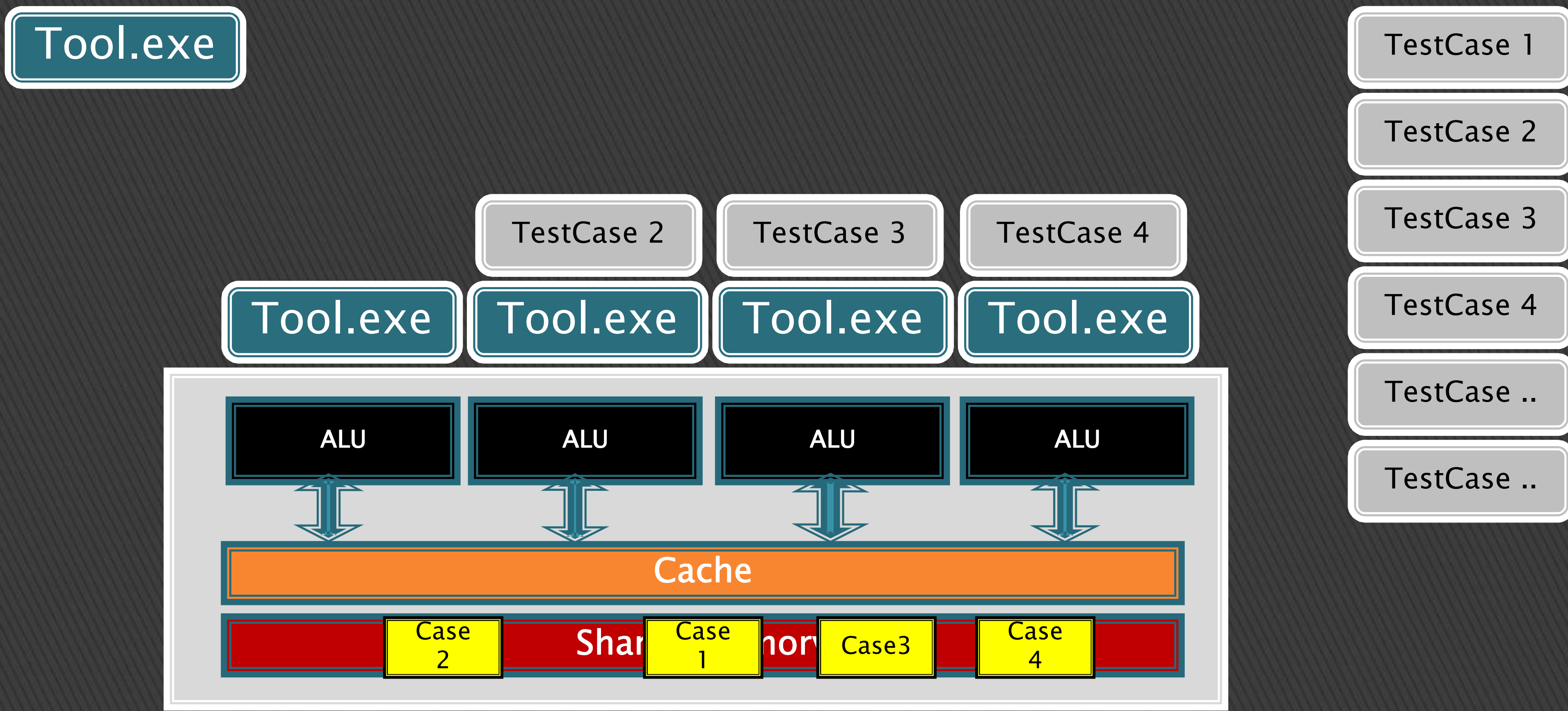
The Need For Parallelization

- The commodity computing environment underwent a paradigm shift in the last 7 years

  Power hungry frequency-scaling replaced by more efficient scaling-in-computing-cores

- "No free-lunch" for software applications

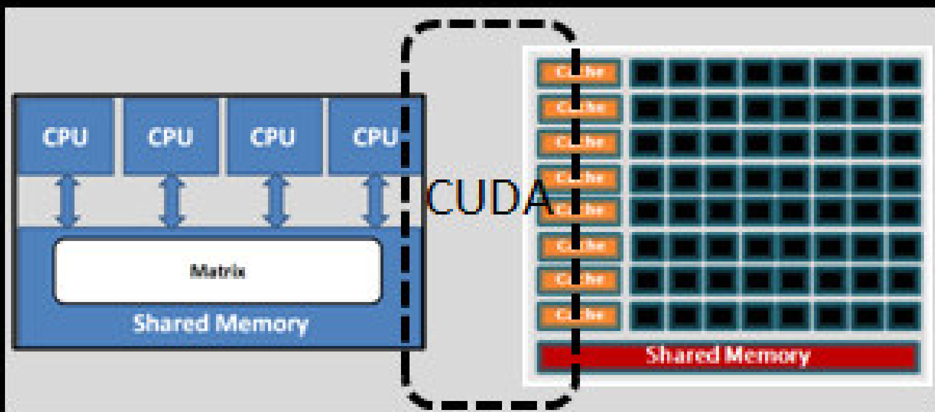
Parallelization: The Easy Way



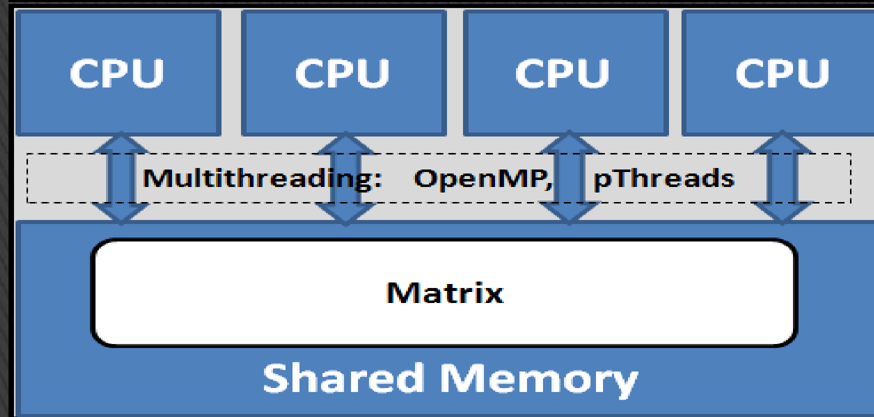
- "Embarrassingly Parallel" Class of Problems
  - Suitable for applications with low memory requirement
- What about memory intensive applications ?
  - Large SPICE runs, Extraction

Parallelization Paradigms

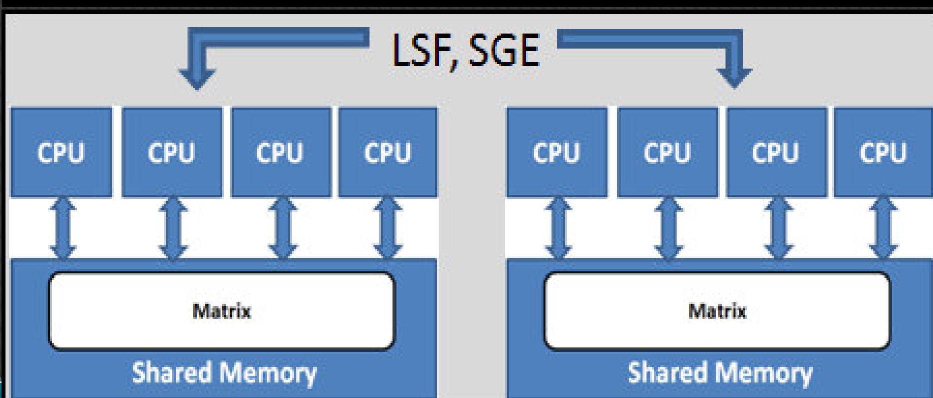
CPU – GPU (FPGA)



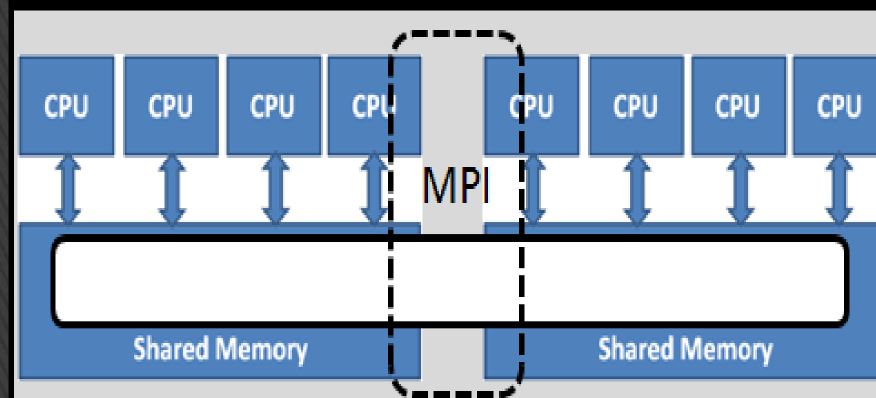
Multithreading



Distribution



MPI



Benefits of Effective Parallelization

Algorithm in which 100% can be efficiently parallelized

Compute Platform	Time	Utility Computing Cost
1 Compute Unit	100 hours	$x
100 Compute Units	1 hours	$x

Algorithm in which 50% can be efficiently parallelized

Compute Platform	Time	Utility Computing Cost
1 Compute Unit	100 hours	$x
100 Compute Units	50 hours	$50x

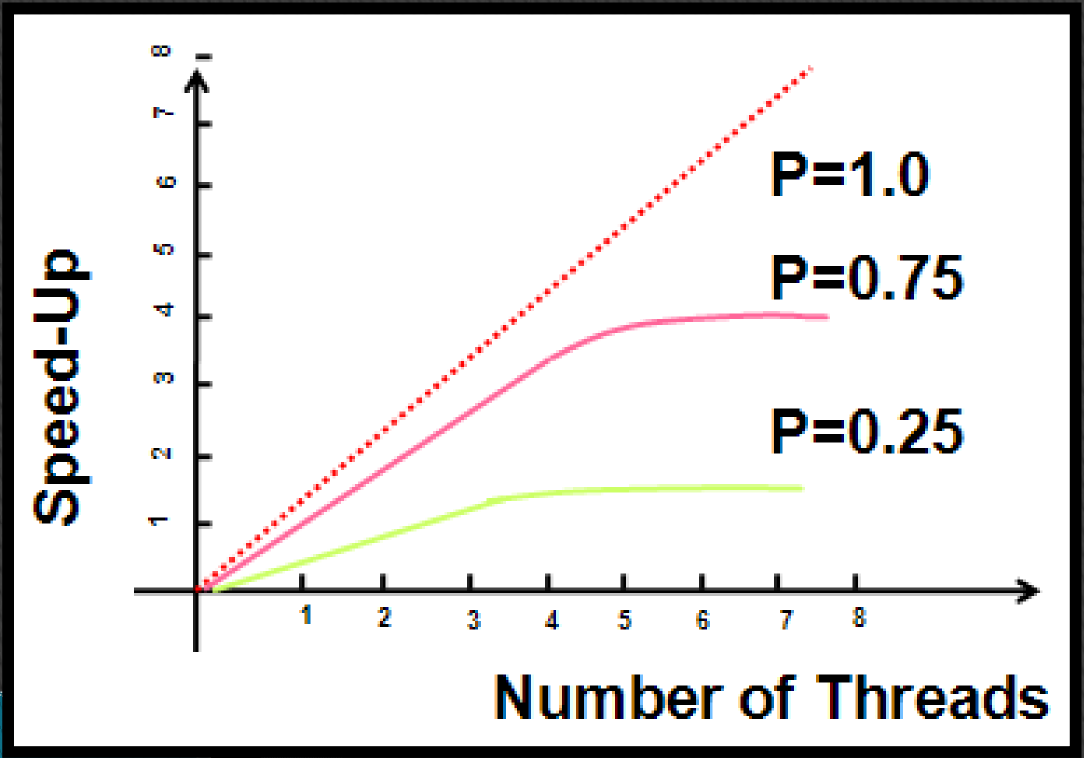
Outline

- ✓ The need for parallelization
- ❑ Challenges towards effective parallelization
- ❑ A multilevel parallelization framework for BEM: A compute intensive application
- ❑ Results and discussion

Amdahl's Law

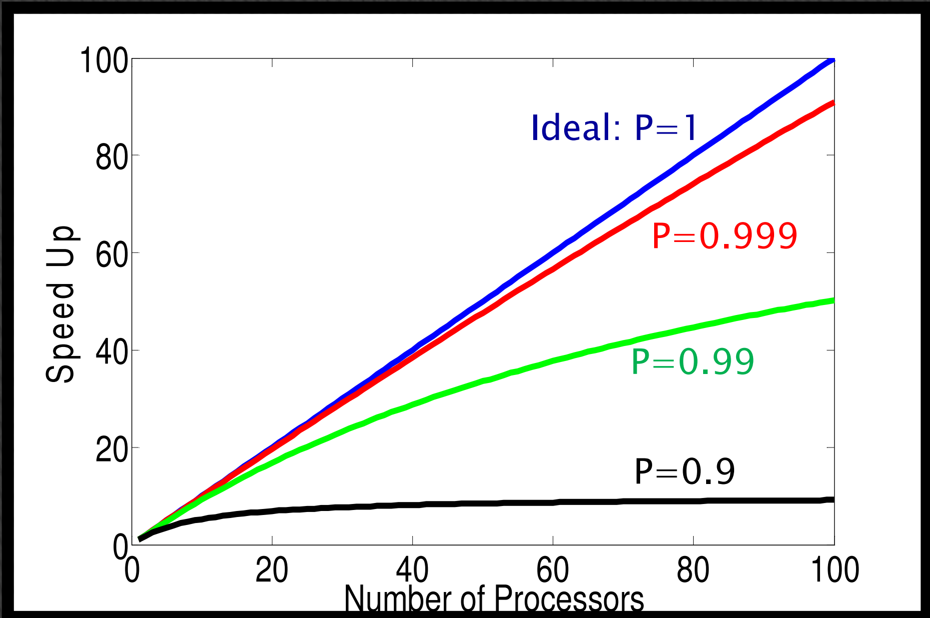
$$\text{Scaling} = \frac{1}{\left[(1-P) + \frac{P}{n}\right]}$$

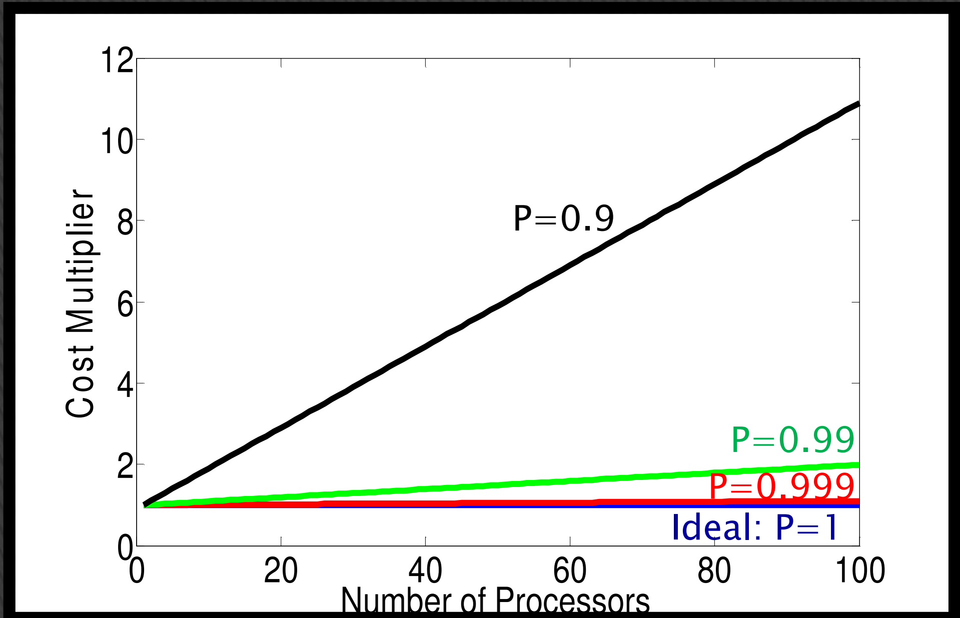
P = Fraction of the algorithm that can be parallelized
(1–P) = Serial content of the algorithm



Parallel performance of an algorithm is severely affected by any serial content

Amdahl's Law: Cloud Perspective





$$\text{Speed Up} = \frac{1}{\left[(1-P) + P/n\right]}$$

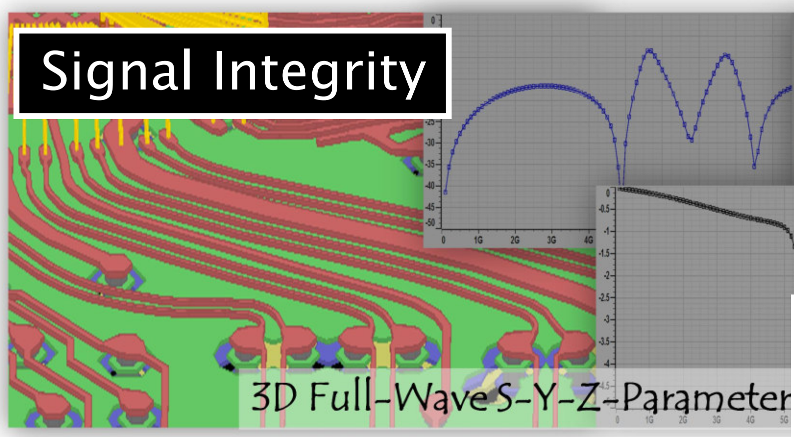
$$\text{Cost Multiplier} = n\left[(1-P) + P/n\right]$$

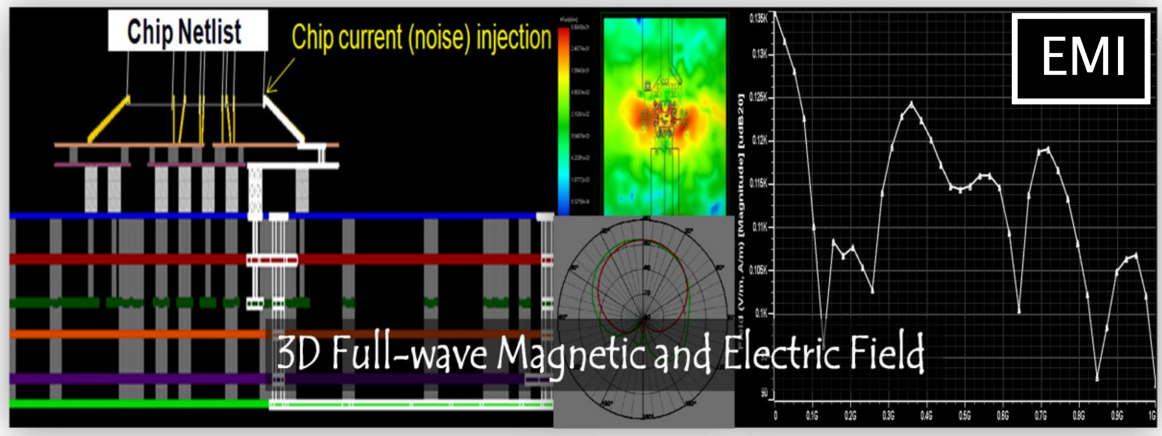
- Power of perfect scaling: Commodity computing cost does not increase even when solving 100x faster

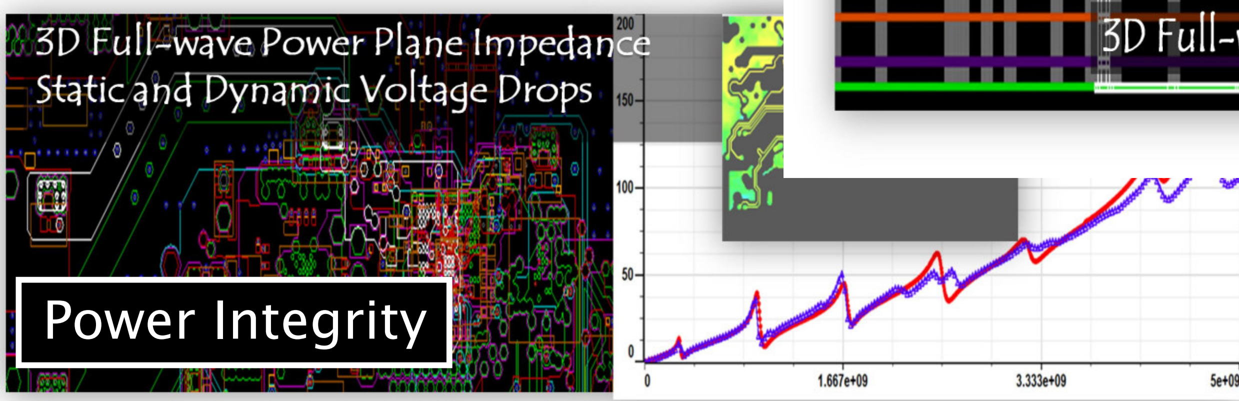
Outline

- ✓ The need for parallelization
- ✓ Challenges towards effective parallelization
- ❑ A multilevel parallelization framework for BEM: A compute intensive application
- ❑ Results and discussion

Application: Large scale problems



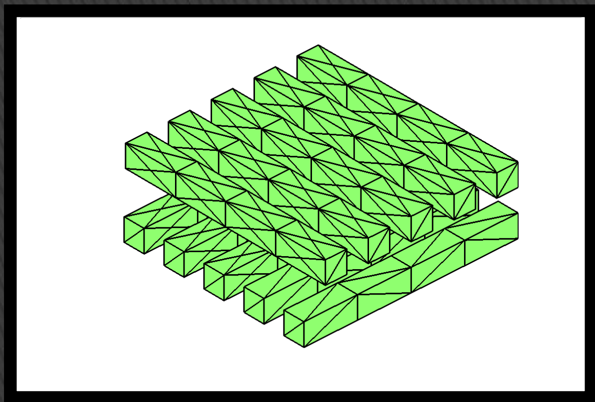


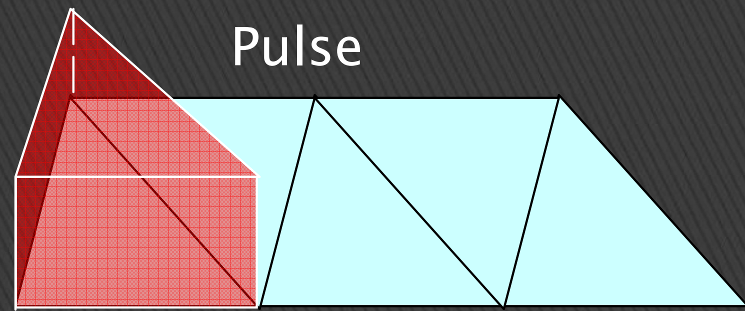


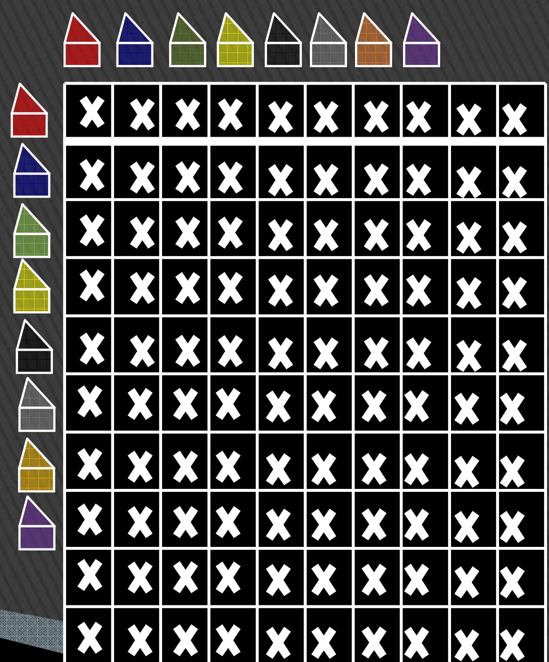
Large Scale Package-Board Analysis

Boundary Element Field Solver

- Surface is discretized into patches (Basis Functions)







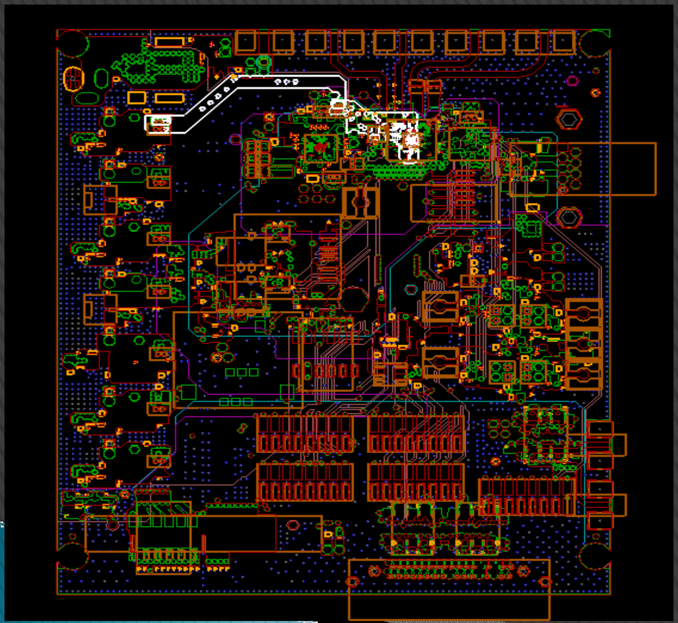
$$\bar{\mathbf{Z}}(j,i) = \left\langle f_j, \int_S G(\mathbf{r},\mathbf{r}')\ f_i(\mathbf{r}')\ ds \right\rangle$$

Dense Method of Moments Matrix

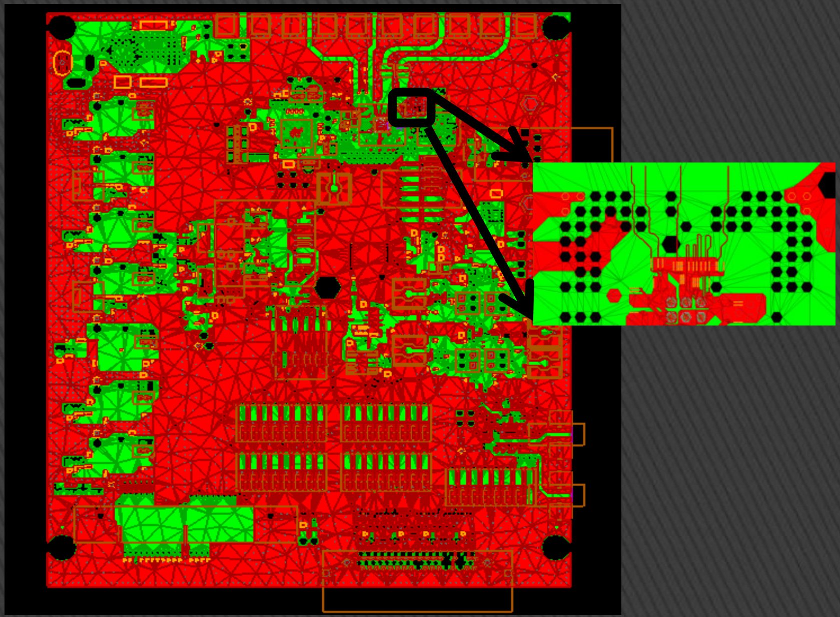
Fast Iterative Matrix Solution

N = Number of basis functions; (150,000)

Scheme	Setup	Solve	Time	Memory
Conventional BEM	$O(N^2)$	$O(N^3)$	2 Days	360 GB
Fast BEM	O(N)	O(N)	20minutes	4 GB

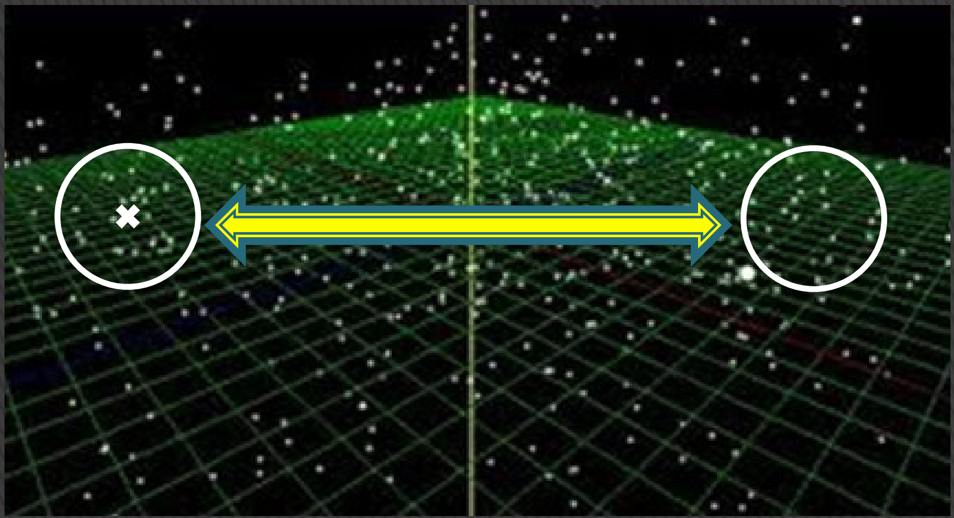


Large problems:
N ~ 1 million



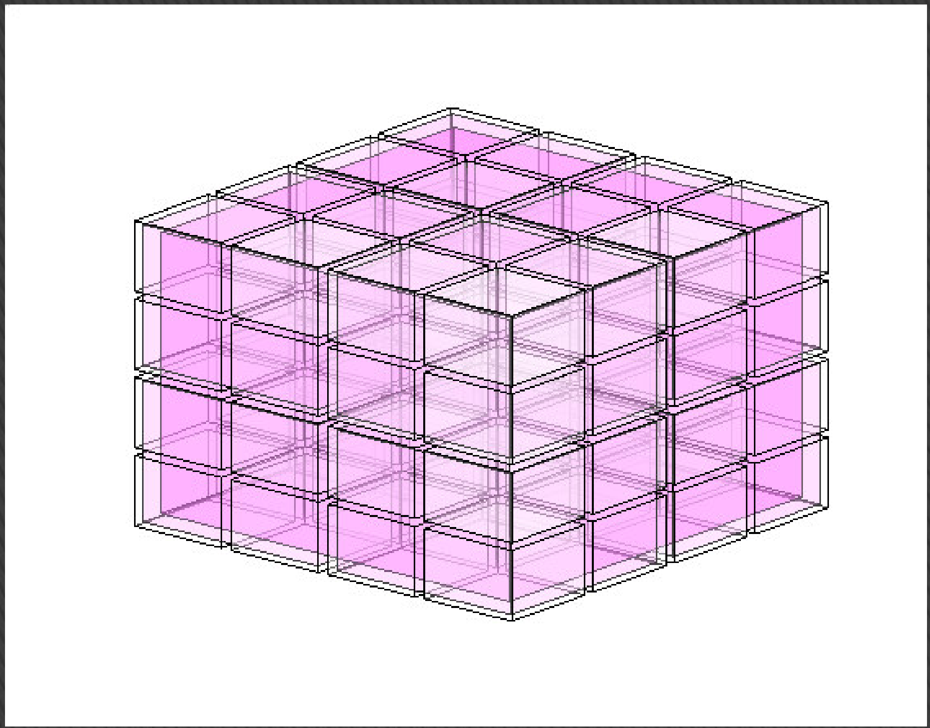
physware™ Design for Integrity

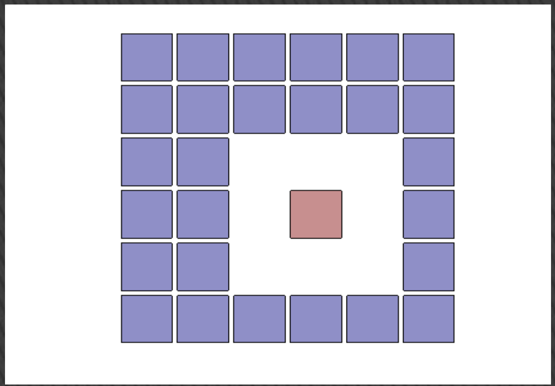
FMM: Fast Solver Physics (1992)



Astronomy Equivalent

Multi-level Algorithm



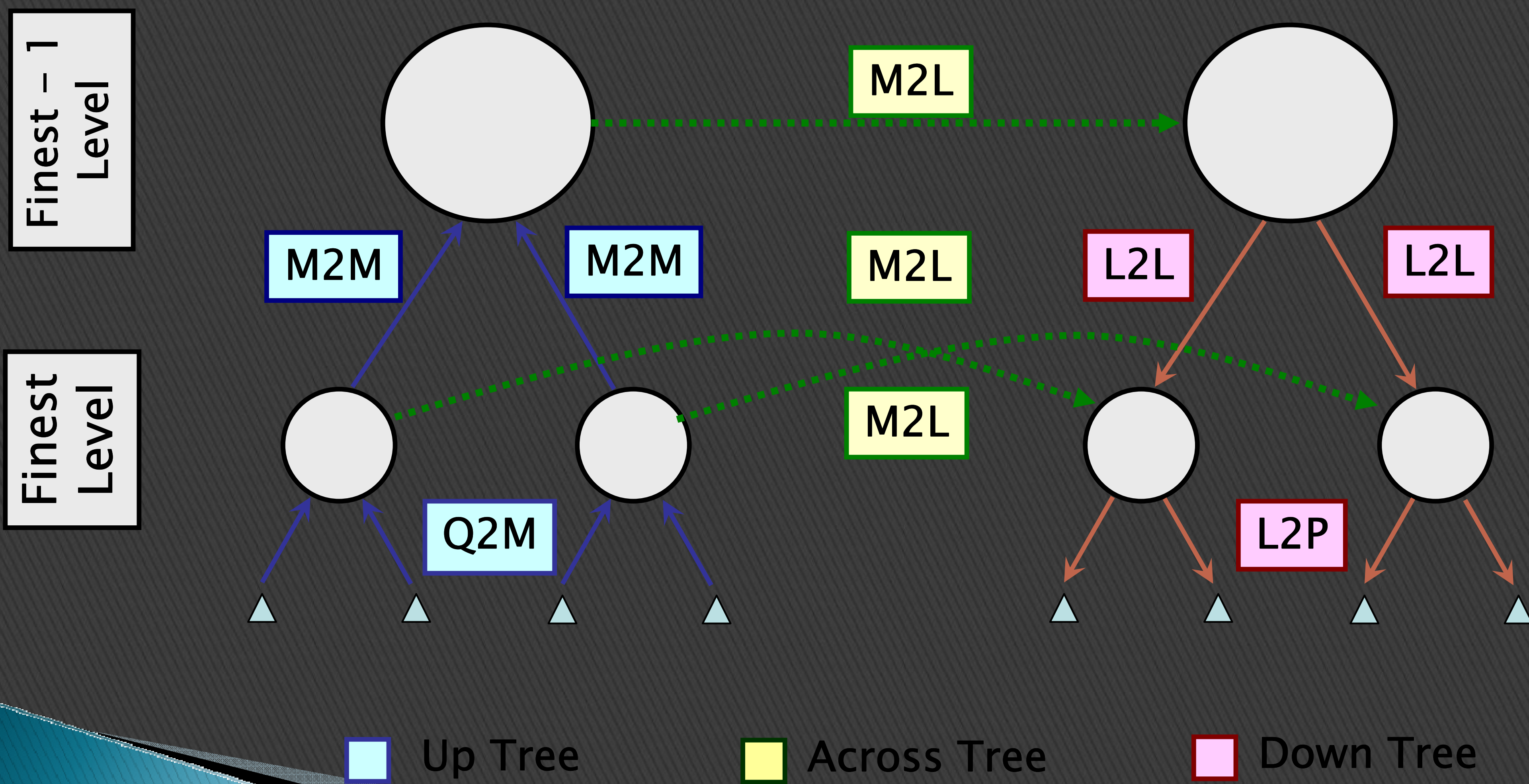


Interaction Shell

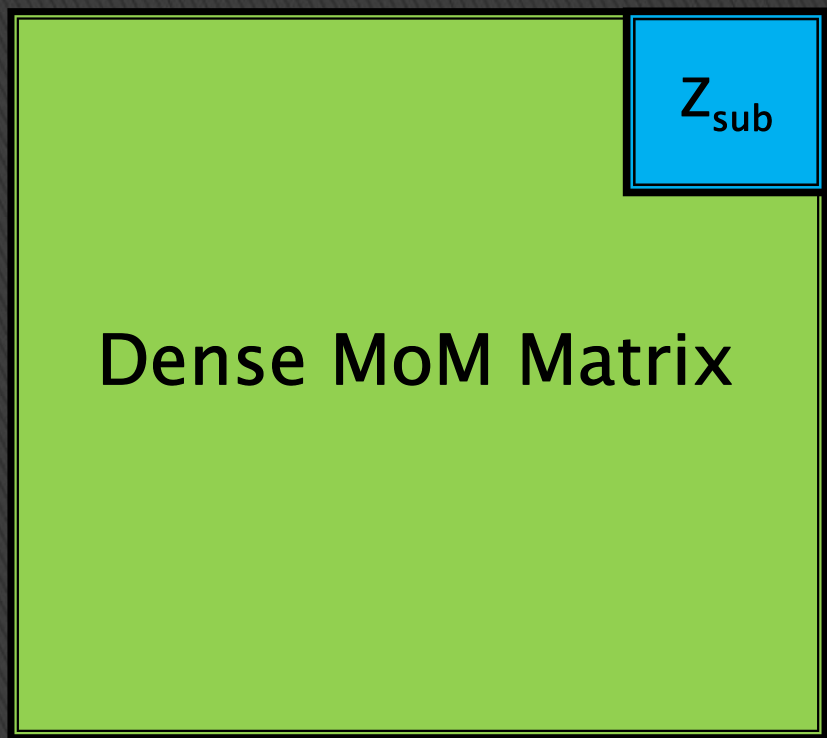
Gravitational Image: Courtesy ParticleMan

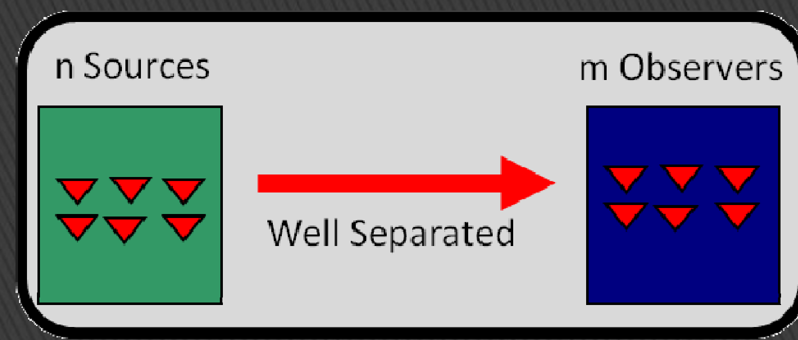
FMM: Fast Solver Algorithm (1992)

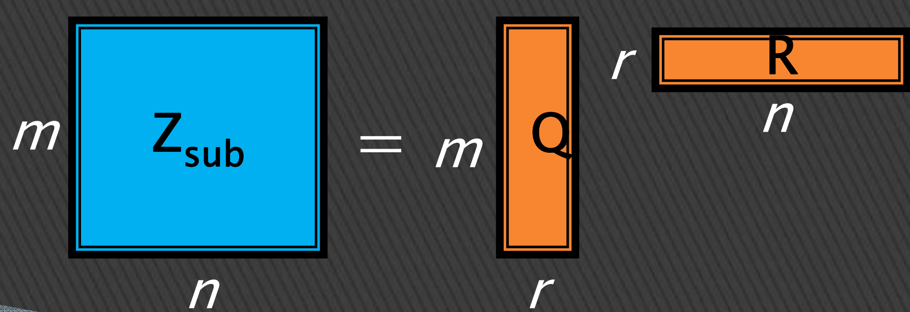
Sequential Algorithm: Impedes Effective Parallelization



SVD/QR Based Compression



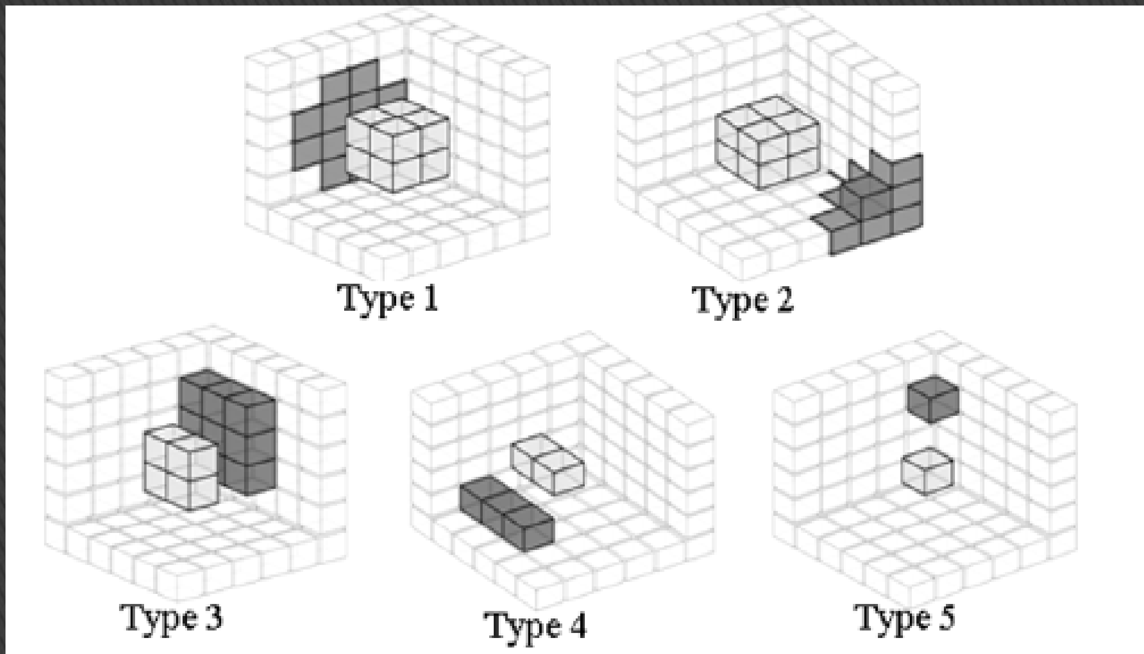


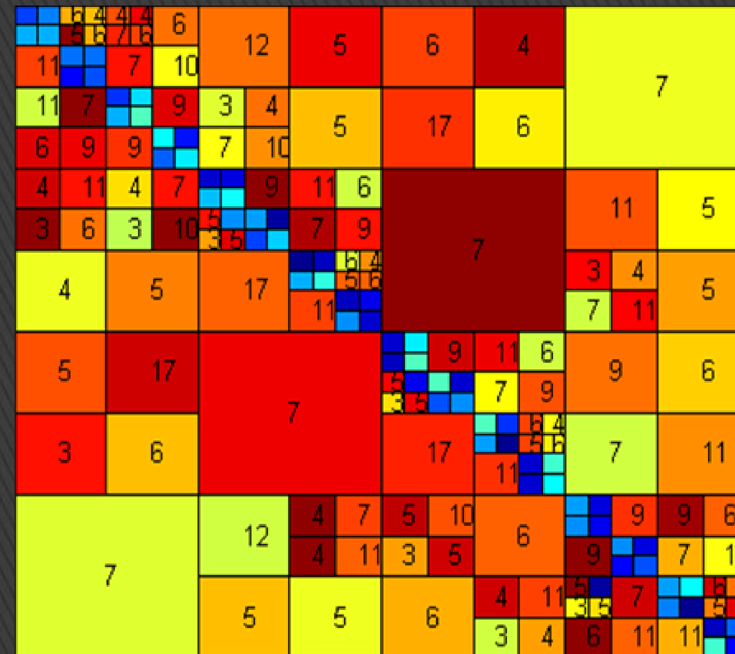


SVD/QR Decomposition Saving

$$\frac{m \times n}{(m+n) \times r}$$

Predetermined Matrix Structure

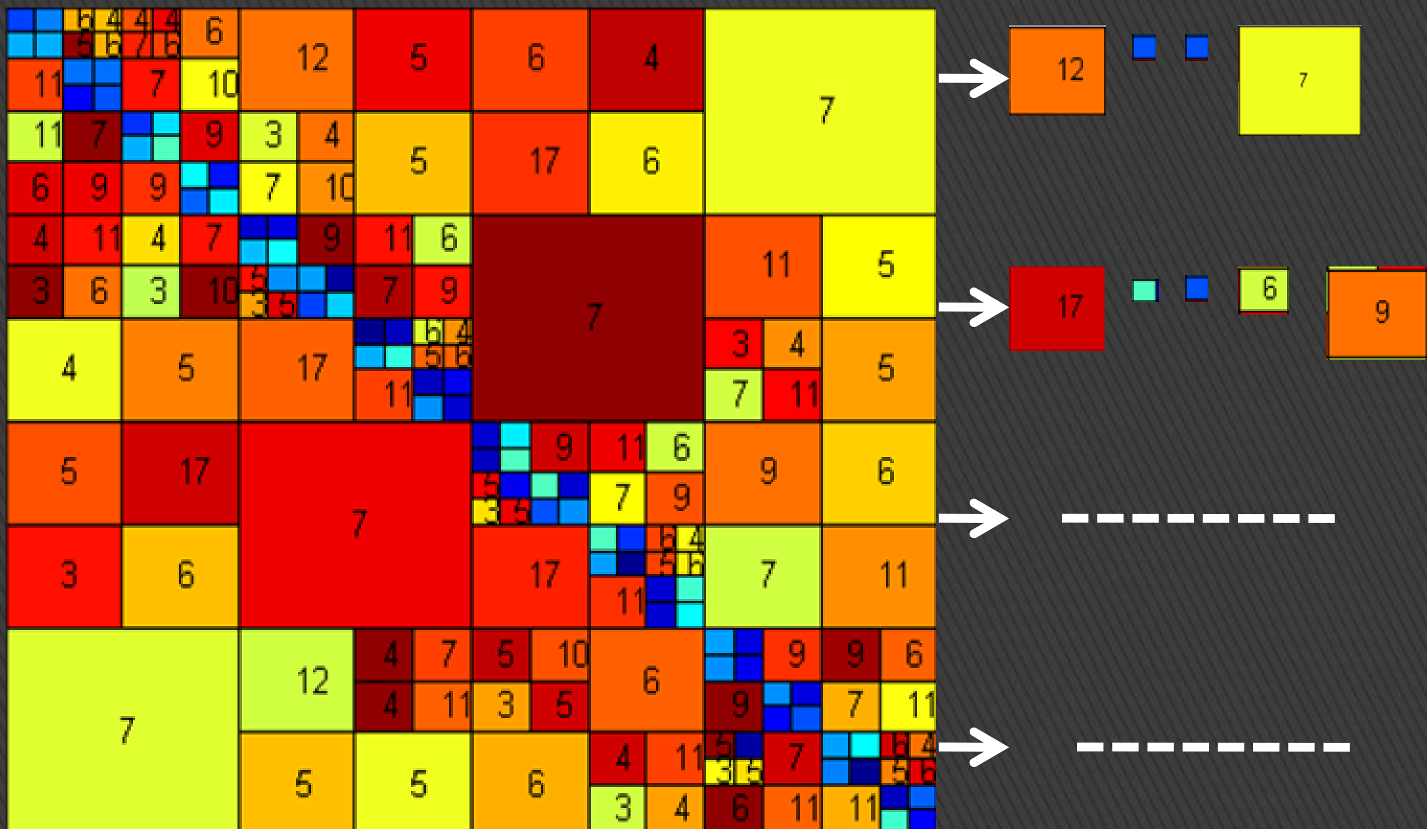


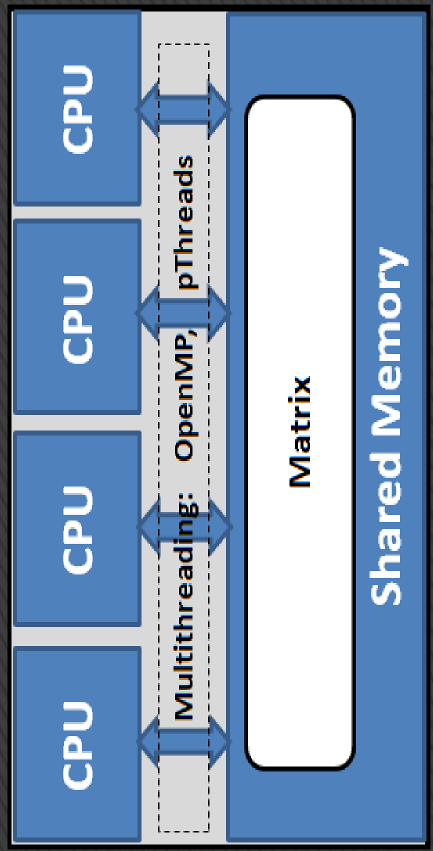


- Pre-determined matrix structure
- Sub-matrix ranks pre-estimated to reasonable accuracy

OpenMP: Matrix Setup and Solve

Predetermined matrix structure given meshed geometry





- ❑ Static load-balancing

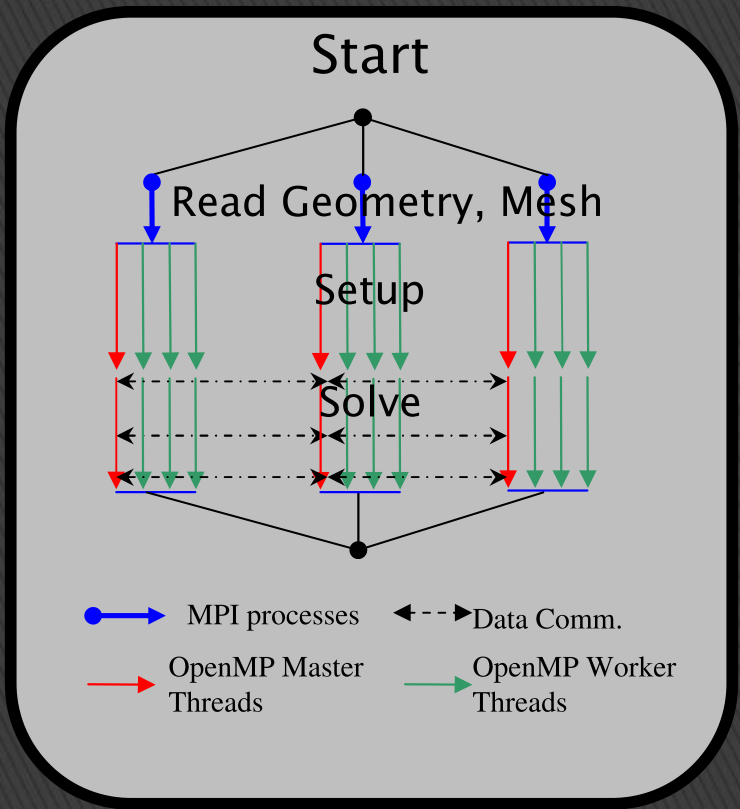
$$N_{avg} = \frac{N}{p} = \sum_{i=1}^{k} \frac{m_i \times n_i}{p}$$

$$F_{avg} = \frac{F}{p} = \frac{1}{p} \sum_{i=1}^{k} (m_i + n_i) \times r_i$$

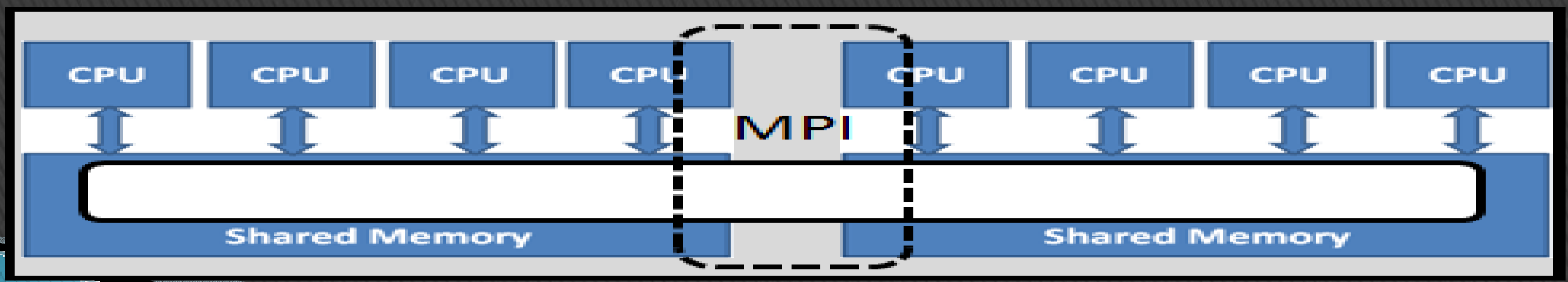
- ❑ Correction by dynamic scheduling

physware™ Design for Integrity

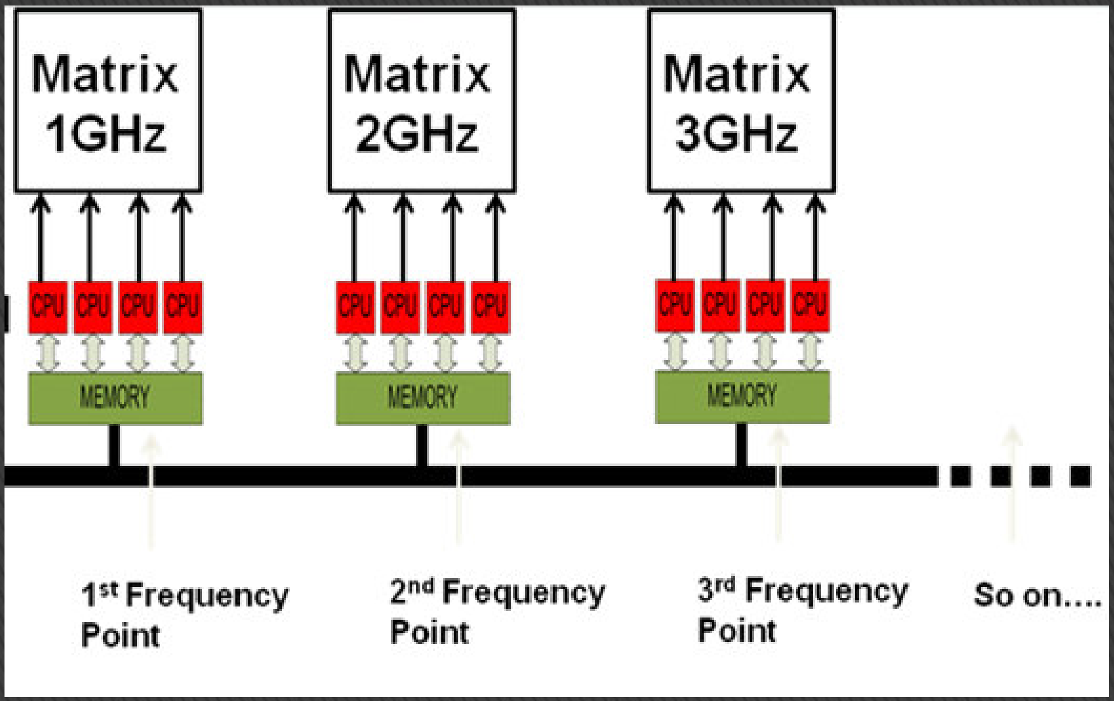
MPI: Matrix Setup and Solve

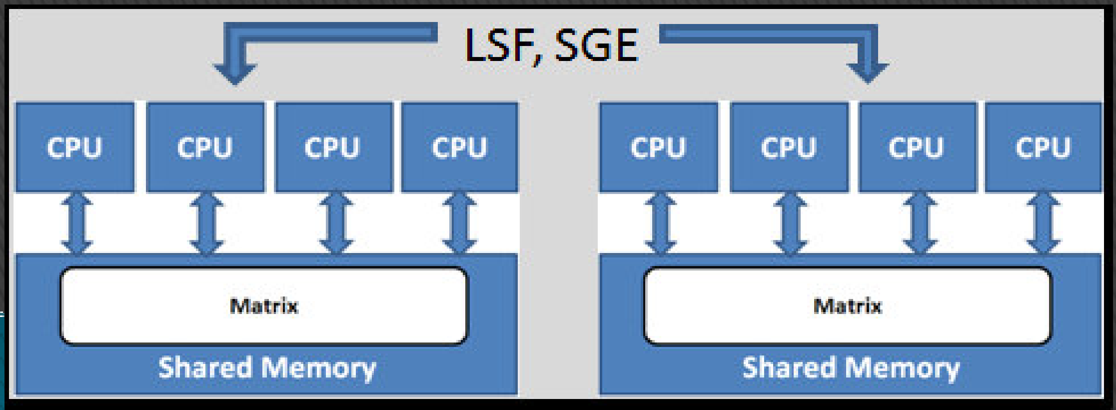


- ❑ All nodes read geometry and mesh
- ❑ Combined DRAM of nodes are used to store the whole matrix, increasing capacity
- ❑ Solve: Vectors are broadcast for matrix-vector operations



SGE: Parallel Frequency Distribution



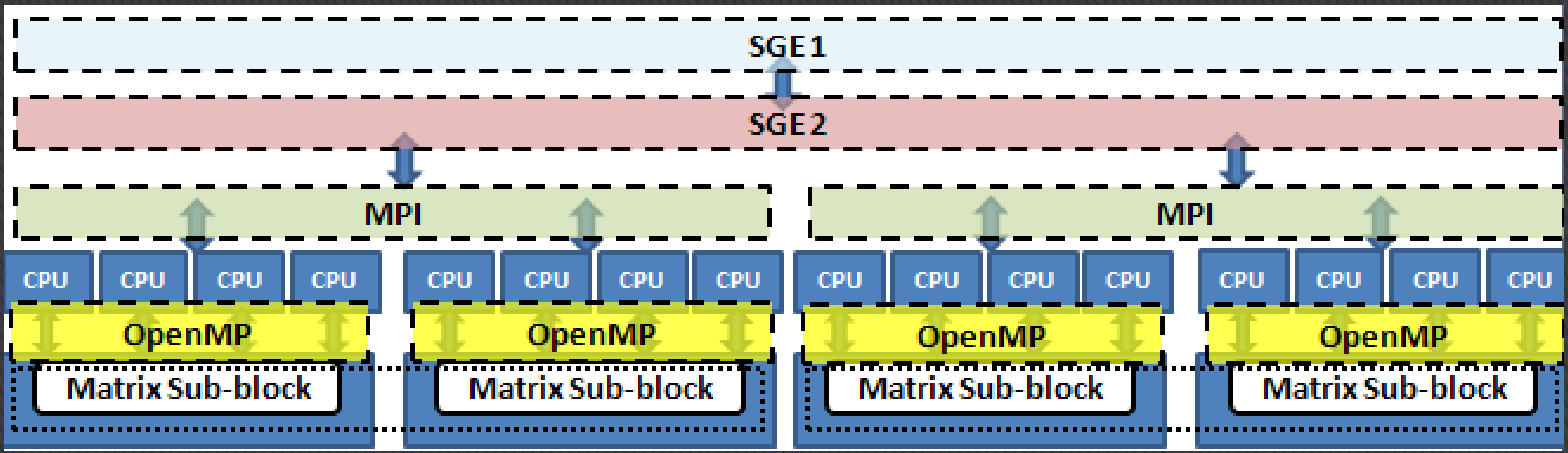


Multiple cores use shared memory utilized to store and compute single frequency matrix

Different frequency matrices on separate node memory

Reveals an embarrassingly parallel level without wasting shared memory

Hybrid Parallel Framework



- ❑ Several layers of parallelism, including embarrassingly parallel layers are employed in a hybrid framework to work around Amdahl's law and achieve high scalability

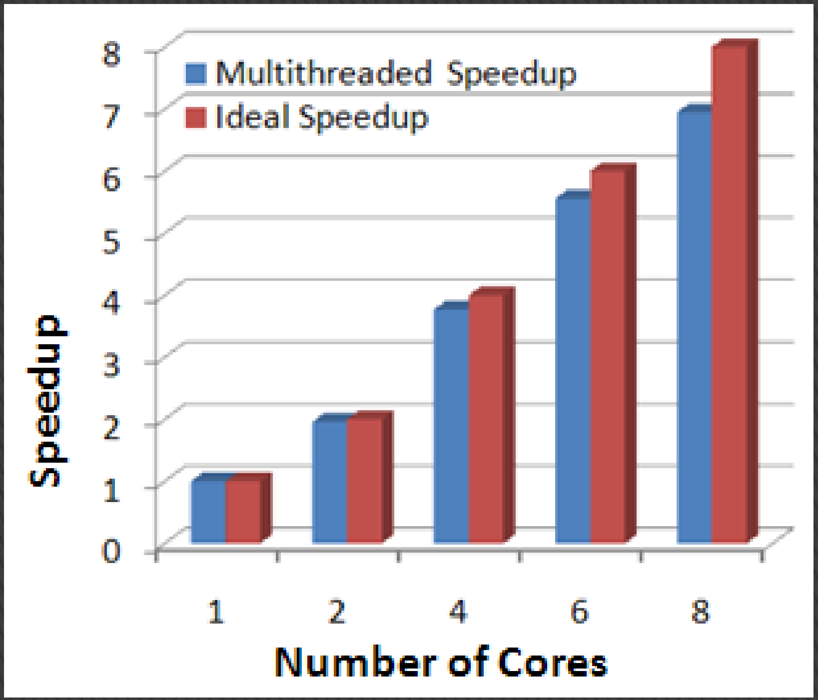
- ❑ Global controller decides the number and type of compute instances to be used at each level

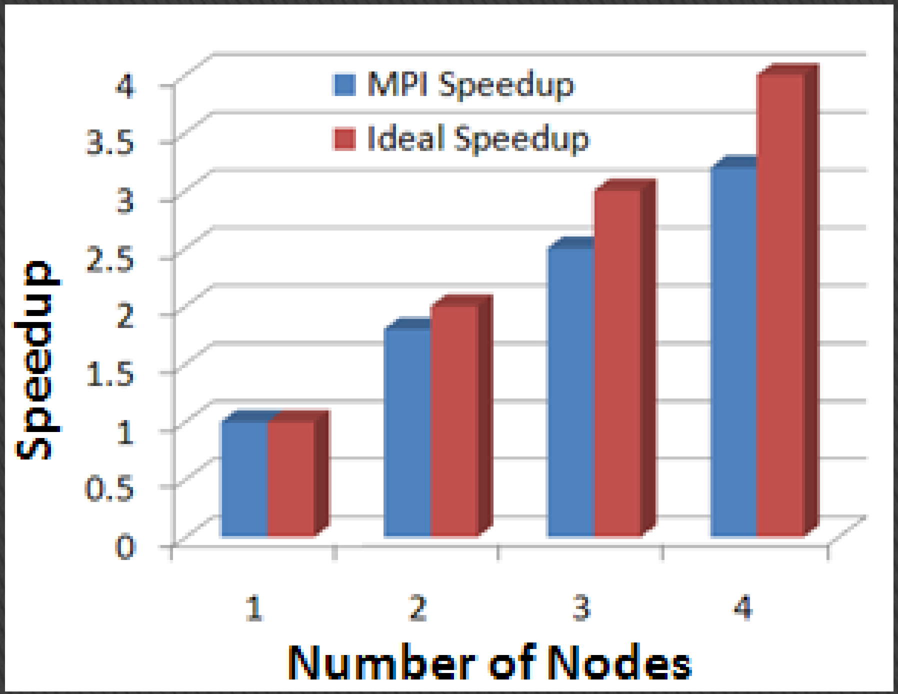
- ❑ File R/W for SGE based parallel layers is performed in binary

Outline

- ✓ The need for parallelization
- ✓ Challenges towards effective parallelization
- ✓ A multilevel parallelization framework for BEM: A compute intensive application
- ❑ Results and discussion

Matrix Setup Analysis

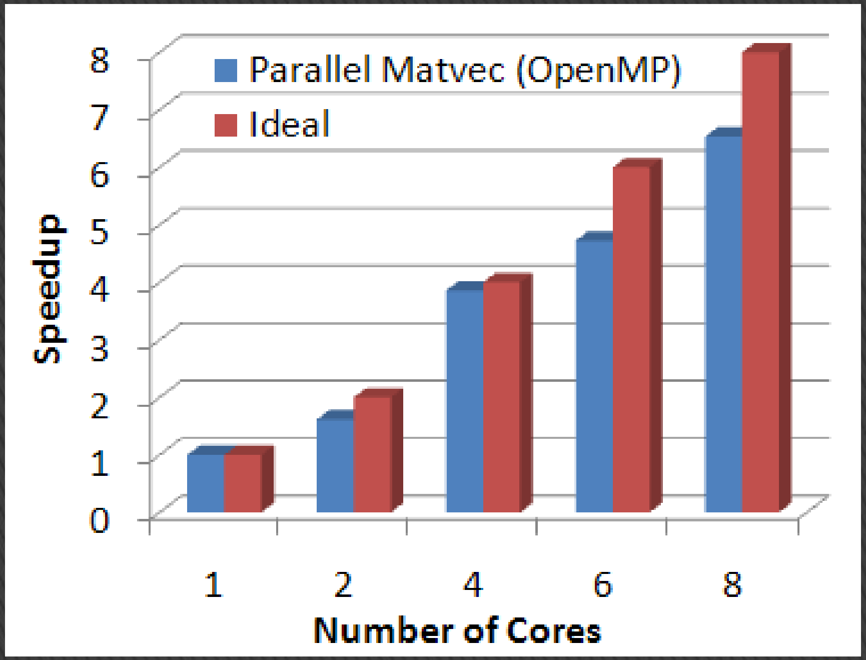


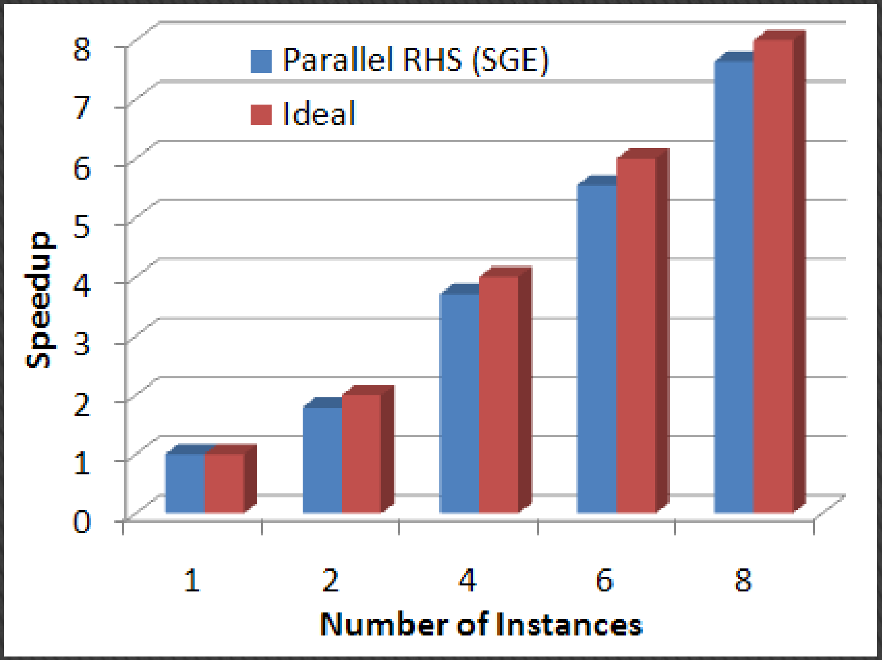


- ❑ Predetermined matrix structure and static-dynamic load balancing achieves close to ideal speed ups

- ❑ Lack of dynamic scheduling across nodes affects MPI speedup

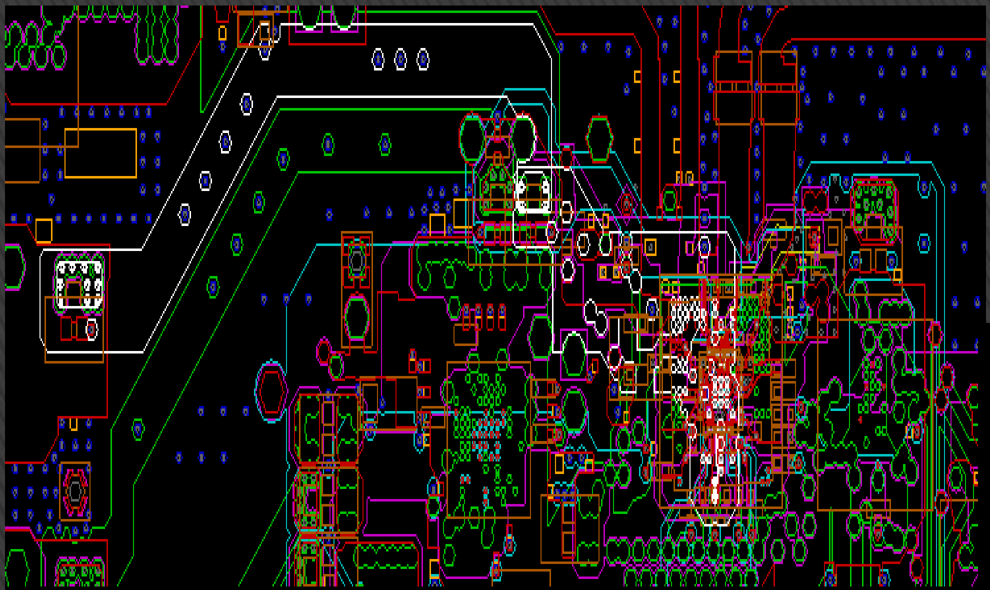
Matrix Solve Analysis

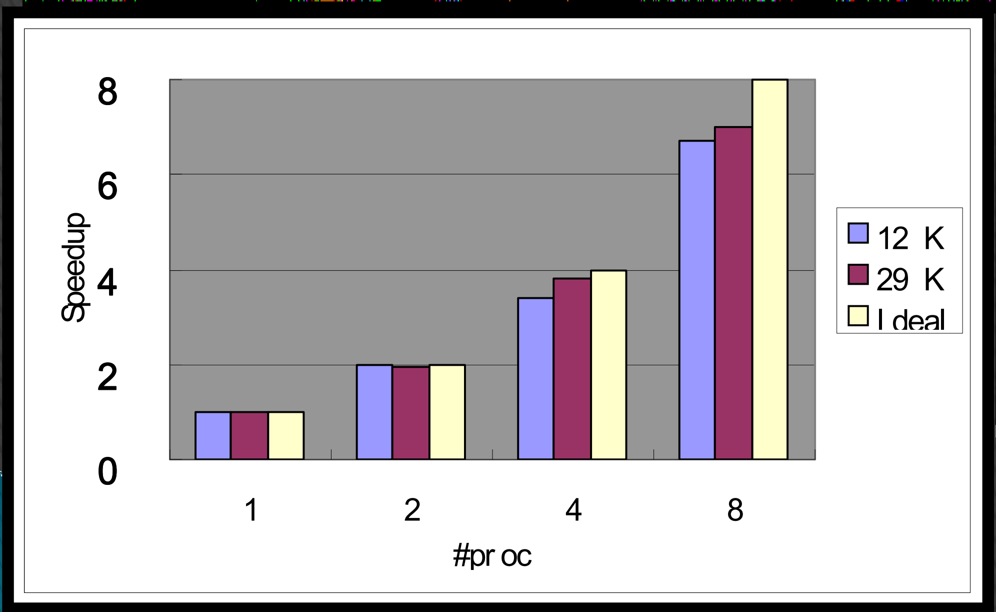


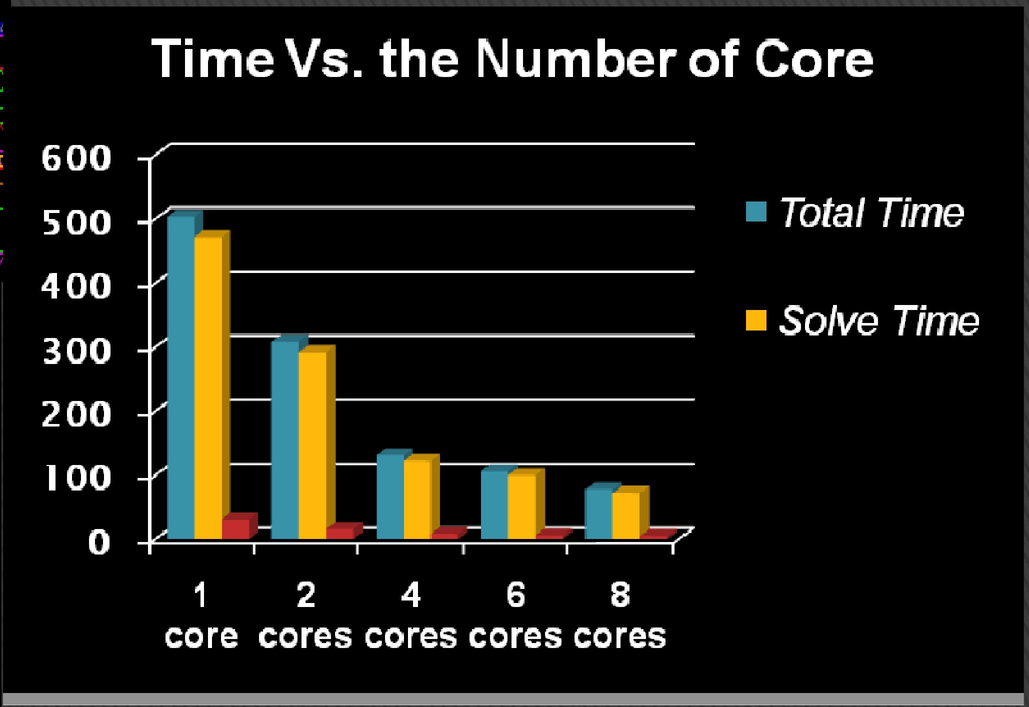


- For large number of right hand sides, SGE based RHS distribution yields better scalability
- Map-reduce after every matrix vector product affects parallel efficiency

Performance on Full-Board Analysis





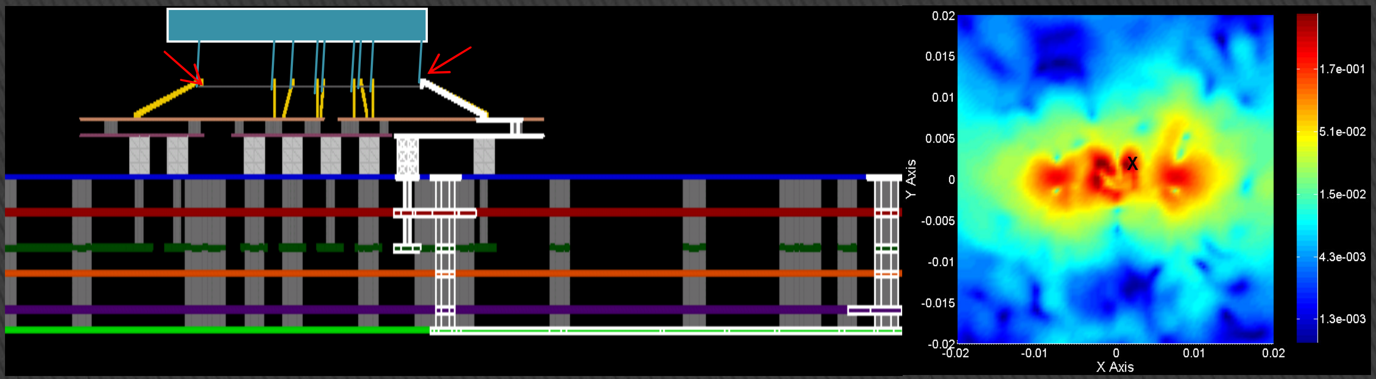


Performance on Amazon EC2

On The Cloud ...

Performance on Amazon EC2

System LSI package and an early design board merged
Meshed with 20,000 basis functions
Broadband frequency simulation for EMI prediction



	Time Taken	Speed-up	Effective Compute Cost
2 core (m1.large)	100 minutes	1	$ 0.60
18 machines with 8 cores (c1.xlarge)	2 minutes	50	$ 0.40

EMI from Cell-Phone Package

Summary

- The future of compute and memory intensive algorithms is "parallel"
- Revisit algorithms from a perspective of multi-processor efficiency as opposed to single-processor complexity
- Even a small serial content can adversely affect the speedup of an algorithm
- Employ levels of parallelism to work-around Amdahl's law
- Presented algorithm achieves around 6.5x speedup on a single 8 core machine and around 50x speed up in a cloud framework using 72 (18x4) cores